Configuration de scripts personnalisés sur CPAR 8.0

Table des matières

Introduction

Conditions préalables

- Exigences
- Composants utilisés

Informations générales

Configurer

- Script Interne Pour Le Trafic Sortant
- Script interne pour le trafic entrant
- Créer un script externe

Introduction

Ce document décrit comment personnaliser le comportement de Cisco Prime Access Registrar (CPAR) 8.0 avec l'utilisation de scripts et de points d'extension.

Conditions préalables

Exigences

Cisco vous recommande de prendre connaissance des rubriques suivantes :

- Administration de CPAR 8.0

Composants utilisés

Les informations contenues dans ce document sont basées sur les versions de matériel et de logiciel suivantes :

- CPAR 8.0 installé sur CentOS 6.5 64 bits

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. Si votre réseau est en ligne, assurez-vous de bien comprendre l’incidence possible des commandes.

Informations générales

CPAR peut être modifié par des scripts internes et externes. Les scripts peuvent être écrits en

C/C++/Java/TCL. Les scripts peuvent être utilisés pour modifier le traitement des paquets RADIUS, TACACS et DIAMETER. Les scripts peuvent être référencés dans CPAR dans les points d'extension. Les points d'extension sont un paramètre/attribut qui apparaît sous certains éléments de configuration et permet de référencer un script. Selon le guide de référence, CPAR n'est pas responsable de toute perte de données, de dommages, etc. causés par des scripts personnalisés.

Voici un exemple de deux points d'extension dans une configuration de périphérique réseau

```
[ //localhost/Radius/Clients/piborowi ]
    Name = piborowi
    Description =
    Protocol = tacacs-and-radius
    IPAddress = 192.168.255.15
    SharedSecret = <encrypted>
    Type = NAS
    Vendor =
    IncomingScript~ =                          // Extension point for incomming traffic
    OutgoingScript~ =                          // Extension point for outgoing traffic
    EnableDynamicAuthorization = FALSE
    NetMask =
    EnableNotifications = FALSE
    EnforceTrafficThrottling = TRUE
```

Selon le guide d'administration de CPAR, plusieurs points d'extension sont disponibles. Un script entrant peut être référencé à chacun de ces points d'extension :

- serveur RADIUS
- Fournisseur (du client immédiat)
- Client (NAS individuel)
- NAS-Vendor-Behind-the-Proxy
- Client-derrière-le-proxy
- Serveur distant (de type RADIUS)
- Service

Un script d'authentification ou d'autorisation peut être référencé à chacun de ces points d'extension :

- Authentification de groupe
- Authentification utilisateur
- Autorisation de groupe
- Autorisation utilisateur

Le script sortant peut être référencé à chacun de ces points d'extension :

- Service
- Client-derrière-le-proxy
- NAS-Vendor-Behind-the-Proxy

- Client (NAS individuel)
- Fournisseur NAS
- serveur RADIUS

Il est essentiel de comprendre l'ordre dans lequel les scripts sont exécutés par CPAR, car il existe plusieurs points d'extension. Reportez-vous au tableau 7-1 du guide de l'administrateur pour voir l'ordre des 29 points de script/extension disponibles.

Un script interne est un script qui est configuré directement dans l'interface de ligne de commande CPAR (aregcmd). Il ne nécessite pas de fichiers externes et beaucoup de connaissances en programmation. Un script externe est un script stocké dans un fichier du système d'exploitation (CENTOS ou RHEL) et simplement référencé dans l'interface de ligne de commande CPAR.

Configurer

Script Interne Pour Le Trafic Sortant

Dans les scripts internes, vous pouvez utiliser les modificateurs suivants :

1. +rsp : - ajoute et attribue à la réponse

2. -rsp : - supprime l'attribut de la réponse

3. #rsp : - remplace l'attribut par une nouvelle valeur

4. ci-dessus peut être utilisé pour req (request/incoming packet et env, qui est le dictionnaire d'environnement). Exemples +req : ou -env :

Ajoutez un script interne sous /Radius/Scripts. Configurez deux AVP supplémentaires à renvoyer avec le paquet d'acceptation d'accès : Filter-Id et Vendor-Specific one (pour rejoindre le domaine vocal).

<#root>

```
-->
ls -R


[ //localhost/Radius/Scripts/addattr ]
    Name = addattr
    Description =
    Language = internal
    Statements/
        1. +rsp:Filter-Id=PhoneACL
        2. +rsp:Cisco-AVPair=device-traffic-class=voice

-->
ls -R


[ Services/local-users ]
```

```
    Name = local-users
    Description =
    Type = local
    IncomingScript~ =
    OutgoingScript~ = addattr
    OutagePolicy~ = RejectAll
    OutageScript~ =
    UserList = Default
    EnableDeviceAccess = True
    DefaultDeviceAccessAction~ = DenyAll
    DeviceAccessRules/
        1. switches
```

Testez avec l'utilisation de radclient local :

<#root>

-->

simple

p011
-->

p011 send

p014
-->

p014

```
Packet: code = Access-Accept, id = 18, length = 64, attributes =
        Filter-Id = PhoneACL
        Cisco-AVPair = device-traffic-class=voice
```

Traces :

```
07/31/2019 10:31:26.254: P2363: Running Service local-users's OutgoingScript: addattr
07/31/2019 10:31:26.254: P2363: Internal Script for 1   +rsp:Filter-Id=PhoneACL : Filter-Id = PhoneACL
07/31/2019 10:31:26.254: P2363: Setting value PhoneACL for attribute Filter-Id
07/31/2019 10:31:26.254: P2363: Trace of Response Dictionary
07/31/2019 10:31:26.254: P2363: Trace of Access-Request packet
07/31/2019 10:31:26.254: P2363:    identifier = 18
07/31/2019 10:31:26.254: P2363:    length = 30
07/31/2019 10:31:26.254: P2363:    respauth = fb:63:14:3f:c1:fb:ac:03:7d:16:29:61:ba:ef:13:4f
```

```
07/31/2019 10:31:26.254: P2363:    Filter-Id = PhoneACL
07/31/2019 10:31:26.254: P2363: Internal Script for 2   +rsp:Cisco-AVPair=device-traffic-class=voice : 
07/31/2019 10:31:26.254: P2363: Setting value device-traffic-class=voice for attribute Cisco-AVPair
07/31/2019 10:31:26.254: P2363: Trace of Response Dictionary
07/31/2019 10:31:26.254: P2363: Trace of Access-Request packet
07/31/2019 10:31:26.254: P2363:    identifier = 18
07/31/2019 10:31:26.254: P2363:    length = 64
07/31/2019 10:31:26.254: P2363:    respauth = fb:63:14:3f:c1:fb:ac:03:7d:16:29:61:ba:ef:13:4f
07/31/2019 10:31:26.254: P2363:    Filter-Id = PhoneACL
07/31/2019 10:31:26.254: P2363:    Cisco-AVPair = device-traffic-class=voice
```

Script interne pour le trafic entrant

Créez un nouveau script qui remplace tous les noms d'utilisateur au format user@domain à anonymous et appliquez-le en tant que script entrant pour le service que vous utilisez.

Configurer:

```
<#root>
-->
cd /Radius/Scripts


-->
add test


-->
set language internal


-->
cd Statements


-->
add 1


-->
cd 1


-->
set statements "#req:User-Name=~(.*)(@[a-z]+.[a-z]+)~\anonymous"
```

```
-->
ls -R


[ //localhost/Radius/Scripts/test ]
    Name = test
    Description =
    Language = internal
    Statements/
        1. #env:User-Name=~(.*)~anonymous

-->
ls -R /Radius/Services/employee-service/


[ /Radius/Services/employee-service ]
    Name = employee-service
    Description =
    Type = local
    IncomingScript~ = test
    OutgoingScript~ =
    OutagePolicy~ = RejectAll
    OutageScript~ =
    UserList = default
    EnableDeviceAccess = FALSE
    DefaultDeviceAccessAction~ = DenyAll
```

Test avec radclient (la requête est très probablement rejetée car le nom d'utilisateur est devenu anonyme) :

```
<#root>
-->
simple




        @cisco.com




p01e
-->
p01e

Packet: code = Access-Request, id = 27, length = 72, attributes =
```

```
User-Name = <username>@cisco.com
User-Password = <password>
NAS-Identifier = localhost
NAS-Port = 7

-->
```

p01e send

```
p020
-->
```

p020

```
Packet: code = Access-Reject, id = 27, length = 35, attributes =
        Reply-Message = Access Denied
```

Trace :

Avant l'exécution de employee-service, trois scripts sont appelés. D'abord, CPAR appelle CiscoIncomingScript, puis il appelle ParseServiceHints qui est attaché à la configuration du client/périphérique réseau localhost. Il extrait le nom d'utilisateur du paquet et le place dans le dictionnaire d'environnement. Deuxième script : le test est appelé et le nom d'utilisateur dans le dictionnaire d'environnement passe de <nom d'utilisateur> à anonyme

client localhost :

```
[ //localhost/Radius/Clients/localhost ]
    Name = localhost
    Description =
    Protocol = radius
    IPAddress = 127.0.0.1
    SharedSecret = <encrypted>
    Type = NAS+Proxy
    Vendor = Cisco
    IncomingScript~ = ParseServiceHints
    OutgoingScript~ =
    EnableDynamicAuthorization = FALSE
    NetMask =
    EnableNotifications = FALSE
    EnforceTrafficThrottling = TRUE
```

Sortie de trace :

<#root>

```
07/31/2019 11:38:53.522: P2855: PolicyEngine: [SelectPolicy] Successful
07/31/2019 11:38:53.522: P2855: Using Client: localhost
07/31/2019 11:38:53.522: P2855: Using Vendor: Cisco
07/31/2019 11:38:53.522: P2855: Running Vendor Cisco's IncomingScript: CiscoIncomingScript
07/31/2019 11:38:53.522: P2855: Running Client localhost IncomingScript: ParseServiceHints
07/31/2019 11:38:53.522: P2855: Rex: environ->get( "Request-Type" ) -> "Access-Request"
07/31/2019 11:38:53.522: P2855: Rex: environ->get( "Request-Type" ) -> "Access-Request"
```

```
07/31/2019 11:38:53.522: P2855: Rex: environ->get( "User-Name" ) ->

"

<username>

"



07/31/2019 11:38:53.522: P2855: Authenticating and Authorizing with Service employee-service
07/31/2019 11:38:53.522: P2855: Running Service employee-service's IncomingScript: test
07/31/2019 11:38:53.522: P2855: Numbered attribute got for the radius / tacacs packet. ignoring # User-
07/31/2019 11:38:53.523: P2855: Numbered attribute got for the radius / tacacs packet. ignoring # User-
07/31/2019 11:38:53.523: P2855: Numbered attribute got for the radius / tacacs packet. ignoring # User-
07/31/2019 11:38:53.523: P2855: Internal Script for 1   #env:User-Name=~(.*)~anonymous : User-Name = an
07/31/2019 11:38:53.523: P2855: Setting value anonymous for attribute User-Name
07/31/2019 11:38:53.523: P2855: Trace of Environment Dictionary
07/31/2019 11:38:53.523: P2855:            User-Name = anonymous
07/31/2019 11:38:53.523: P2855:            NAS-Name-And-IPAddress = localhost (127.0.0.1)
07/31/2019 11:38:53.523: P2855:            Authorization-Service = employee-service
07/31/2019 11:38:53.523: P2855:            Source-Port = 51169
07/31/2019 11:38:53.523: P2855:            Authentication-Service = employee-service
07/31/2019 11:38:53.523: P2855:            Trace-Level = 1000
07/31/2019 11:38:53.523: P2855:            Destination-Port = 1812
07/31/2019 11:38:53.523: P2855:            Destination-IP-Address = 127.0.0.1
07/31/2019 11:38:53.523: P2855:            Source-IP-Address = 127.0.0.1
07/31/2019 11:38:53.523: P2855:            Enforce-Traffic-Throttling = TRUE
07/31/2019 11:38:53.523: P2855:            Request-Type = Access-Request
07/31/2019 11:38:53.523: P2855:            Script-Level = 6
07/31/2019 11:38:53.523: P2855:            Provider-Identifier = Default
07/31/2019 11:38:53.523: P2855:            Request-Authenticator = 5f:62:5a:72:0f:7b:a2:2a:9c:06:ba:2e:
07/31/2019 11:38:53.523: P2855:            Realm = cisco.com
07/31/2019 11:38:53.523: P2855: Getting User anonymous's UserRecord from UserList Default
07/31/2019 11:38:53.523: P2855: Failed to get User anonymous's UserRecord from UserList Default
07/31/2019 11:38:53.523: P2855: Running Vendor Cisco's OutgoingScript: CiscoOutgoingScript
07/31/2019 11:38:53.523: P2855: Trace of Access-Reject packet
07/31/2019 11:38:53.523: P2855:    identifier = 27
07/31/2019 11:38:53.523: P2855:    length = 35
07/31/2019 11:38:53.523: P2855:    respauth = d3:7d:b3:f6:05:47:2c:66:d9:c0:01:7d:67:d7:93:99
07/31/2019 11:38:53.523: P2855:    Reply-Message = Access Denied
07/31/2019 11:38:53.523: P2855: Sending response to 127.0.0.1
```

Créer un script externe

Ajoutez un fichier nadip.tcl au répertoire /opt/CSCOar/scripts/radius/tcl/ et ajoutez ce contenu :

<#root>

```
[root@piborowi-cpar80-16 tcl]#

cat /opt/CSCOar/scripts/radius/tcl/nadip.tcl


proc UpdateNASIP {request response environ} {
 $request trace 2 "TCL CUSTOM_SCRIPT Updating NAS IP ADDRESS"
 $request trace 2 "Before put: " [ $request get NAS-IP-Address ]
 $request put NAS-IP-Address 1.2.3.4
 $request trace 2 "After put: " [ $request get NAS-IP-Address ]
```

```
}
```

Contenu de nadip.tcl expliqué ligne par ligne :

Ligne #1 Définition et arguments de la procédure. Demandez, répondez, et trois dictionnaires disponibles où vous pouvez modifier les données de session/paquet.

Ligne #2 Ligne de débogage pour le script à imprimer comme niveau de trace 2.

Line #3 Content of NAS-IP-Address attribute in request dictionary avant de définir cette valeur.

Line #4 Définissez l'attribut Nas-IP-Address dans le dictionnaire de requête sur la valeur 1.2.3.4.

Line #5 Imprimez à nouveau l'attribut NAS-IP-Address.

Une fois le script créé et enregistré dans le système d'exploitation, configurez la référence CPAR au script. Définissez la langue comme TCL, le nom de fichier doit être le nom de fichier exact avec l'extension (dans ce cas, il s'agit de nadip.tcl). EntryPoint est le nom de la procédure du fichier que vous souhaitez exécuter en tant que script. Référence a créé un script CPAR sous service (incomingScript) et testé avec radclient.

Les lignes #2, #3, #5 peuvent être observées dans la trace :

<#root>

```
-->

ls -R /Radius/scripts/nadipaddress/


[ /Radius/Scripts/nadipaddress ]
    Name = nadipaddress
    Description =
    Language = tcl              <<<<<<<<
    Filename = nadip.tcl        <<<<<<<<
    EntryPoint = UpdateNASIP    <<<<<<<<
    InitEntryPoint =
    InitEntryPointArgs =

-->

ls -R /Radius/services/employee-service/


[ /Radius/Services/employee-service ]
    Name = employee-service
    Description =
    Type = local
    IncomingScript~ = nadipaddress    <<<<<<<<
    OutgoingScript~ =
    OutagePolicy~ = RejectAll
    OutageScript~ =
    UserList = default
    EnableDeviceAccess = FALSE
```

```
    DefaultDeviceAccessAction~ = DenyAll
```

Trace :

```
07/31/2019 13:40:53.615: P3490: Running Service employee-service's IncomingScript: nadipaddress
07/31/2019 13:40:53.615: P3490: TCL CUSTOM_SCRIPT Updating NAS IP ADDRESS
07/31/2019 13:40:53.616: P3490:     Tcl: request trace 2 TCL CUSTOM_SCRIPT Updating NAS IP ADDRESS -> O
07/31/2019 13:40:53.616: P3490:     Tcl: request get NAS-IP-Address -> <empty>
07/31/2019 13:40:53.616: P3490: Before put:
07/31/2019 13:40:53.616: P3490:     Tcl: request trace 2 Before put:   -> OK
07/31/2019 13:40:53.616: P3490:     Tcl: request put NAS-IP-Address 1.2.3.4 -> OK
07/31/2019 13:40:53.616: P3490:     Tcl: request get NAS-IP-Address -> 1.2.3.4
07/31/2019 13:40:53.616: P3490: After put: 1.2.3.4
07/31/2019 13:40:53.616: P3490:     Tcl: request trace 2 After put:  1.2.3.4 -> OK
```

À propos de cette traduction
Cisco a traduit ce document en traduction automatisée vérifiée par une personne dans le cadre d'un service mondial permettant à nos utilisateurs d'obtenir le contenu d'assistance dans leur propre langue.

Il convient cependant de noter que même la meilleure traduction automatisée ne sera pas aussi précise que celle fournie par un traducteur professionnel.